ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

INGENIERO TÉCNICO DE TELECOMUNICACIÓN, ESPECIALIDAD EN SONIDO E IMAGEN

TECNOLOGÍAS DE INTERACCIÓN AVANZADA Y REALIDAD AUMENTADA

Imanol Zabalza Grau

Iosu Azkue Odriozola

Pamplona, 17 de Junio del 2011

ÍNDICE

PAG.

1.- INTRODUCCIÓN

Como bien avanza el título, el objetivo de este proyecto no va a ser el desarrollo completo de una aplicación de interacción avanzada o de realidad aumentada, si no que consistirá en el estudio y comprensión de algunas de las ya existentes.

Se entiende que este proyecto va dirigido a personas con unos conocimientos básicos en temas de programación, Internet, interacción entre periféricos y comunicación entre estos puesto que en este se utilizará un leguaje orientado a estos campos.

Entre las tecnologías a tratar podemos distinguir dos grupos, las que son realizadas mediante software libre y medios de bajo coste, y aquellas que han desarrollado las grandes empresas para su uso comercial.

En los diferentes capítulos se intentará explicar el funcionamiento de algunas de las aplicaciones, que de una manera lo más simple posible, podemos modificar y adaptar a nuestras necesidades para así poder desarrollar nuestra propia aplicación de realidad aumentada ó de interacción.

En primer lugar, se comenzará por el estudio de las posibilidades que nos aporta el mando de control de la consola de Nintendo Wii y su interacción sobre una computadora.

En segundo lugar, veremos una de las posibilidades actuales de desarrollar nuestra propia aplicación de realidad aumentada, hecha a nuestro gusto.

Se debe de tener en cuenta que la mayoría de las pruebas se han realizado con el sistema operativo Mac OS Leopard, y más concretamente con un equipo Macbook White, cuyas características son:

- MAC OS Leopard versión 10.5.8
- Procesador Core 2 Duo 2GHz
- 2 GB SDRAM 667 MHz
- 3MB caché nivel 2

2.- ESTADO DEL ARTE

En la actualidad, existen varios grupos de desarrollo que investigan nuevas aplicaciones e intentan desarrollar nuevas interfaces en este ámbito. Incluso alguna de ellas ya se esta comercializando. A continuación se expondrán algunas de ellas.

2.1.Kinect

Kinect, inicialmente conocido como "Project Natal", es un controlador de juego libre y entretenimiento desarrollado por Microsoft para la videoconsola Xbox 360. La particularidad de este periférico, es la posibilidad de interacción entre videoconsola y usuario sin necesidad de tener contacto físico con ningún controlador de videojuegos.

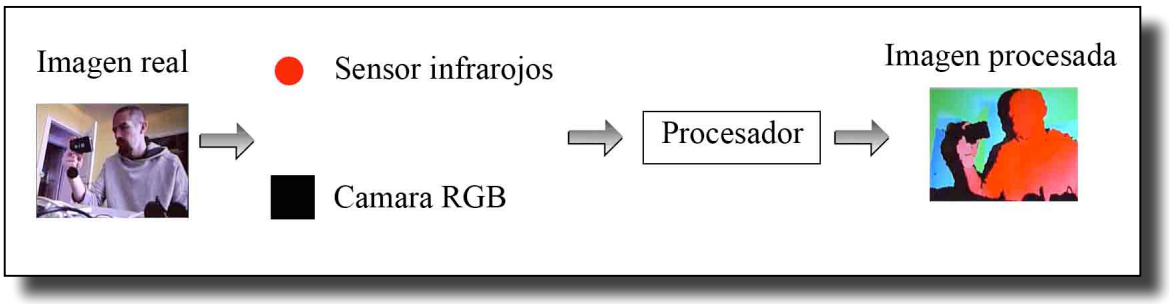
Para ello, utiliza cuatro elementos:

-Cámara RGB: Analiza los tres colores básicos permitiendo el reconocimiento facial con una calidad de 640×480 32-bit a 30 fps.

-Sensor de profundidad: Mediante la proyección de un láser infrarrojo por toda el área donde se encuentra el dispositivo, un sensor CMOS monocromo permite a Kinect "ver" el entorno en tres dimensiones en cualquier condición de luz con una calidad de imagen de 320×240 16-bit a 30 fps.

-Micrófono multidireccional: Mediante un array de micrófonos colocados los laterales del dispositivo, es capaz de localizar voces y sonidos en el entorno 3D, permitiendo la eliminación de ruido localizado.

-Procesador interno: Éste, ejecuta un software patentado que es capaz de analizar el cuerpo completo, extraer cualquier tipo de interferencia alrededor, además de ser el responsable del reconocimiento facial y de voz.



2.1.1 Sistema de rastreo

Este cuenta con un emisor de infrarrojos y dos cámaras.

El emisor de infrarrojos llena la sala de cientos de puntos. Los píxeles que Kinect recibe como ruido en el receptor de infrarrojos son convertidos en una escala de colores, haciendo que los cuerpos, dependiendo de la distancia, se capturen como rojos, verdes, azules hasta llegar a tonos grises, que representan a objetos muy lejanos. El software toma estas imágenes y las hace pasar por una serie de filtros para que Kinect determine qué es una persona y qué no lo es. De esta manera se procesa la imagen buscando patrones, como pueden ser el tamaño del cuerpo humano, cantidad de extremidades que este tiene, etc. Mediante este proceso es capaz de distinguir incluso entre hombro derecho y hombro izquierdo. Todo ello para evitar que objetos del lugar donde nos encontramos sean reconocidos como otros jugadores.

Una vez que la información es separada del resto, se convierte cada identificación del cuerpo en un esqueleto con articulaciones móviles. Además, está precargado con una base de datos de 200 poses, para llenar los espacios en caso de hacer un movimiento que obstruya la visión de la cámara (como echar los brazos hacia atrás).

El sistema completo es capaz de seguir hasta a 6 personas con 2 jugadores activos, y monitorizar 20 articulaciones por cada jugador.

Como factor negativo, nos encontramos que las manos se toman como un objeto agrupado en lugar de capturar los dedos por separado.

2.1.2 Sistema de sonido

En principio, el sistema está preparado para el reconocimiento de voz y en control por comandos de voz, aunque ésta opción en España aparece desactivada hasta una nueva actualización de software.

El sistema de sonido está formado por: tres micrófonos en la parte izquierda y uno en la parte derecha.

Con el objetivo de corregir la señal recibida y que ésta no se vea afectada por ruidos o incluso por el propio sistema de sonido de la Xbox, éste determina aproximadamente la posición de la fuente de sonido, utilizando la diferencia de fases con la que llegan los sonidos a los diferentes micrófonos. Una vez calculada la procedencia del sonido, mediante un complejo algoritmo se combinan las señales de todos los micrófonos, obteniendo una señal que contiene el sonido que llega desde un cono imaginario que parte del dispositivo y que se expande hacia nosotros.

Puesto que éste sistema va a ser usado como reconocimiento de voz, lo primero que se corrige son las frecuencias a muestrear, que serán filtradas entre 80 y 1100 Hz (frecuencias entre las cuales se sitúa la voz humana).

Además, cuando calibramos el micrófono, éste lo hace según la reverberación del entorno, y así poder filtrar los ecos producidos por el rebote del sonido sobre los objetos de

la sala, de manera que si cambiamos de posición los muebles, tendremos que calibrar de nuevo el micrófono.

2.1.3 Sofware libre

En noviembre de 2010, Industrias Adafruit ofreció una recompensa para un controlador de código abierto para Kinect. El 10 de noviembre de ese mismo año, se anunció al español Héctor Martín como el ganador, que en dos horas usó métodos de ingeniería inversa con Kinect y desarrolló un controlador para GNU/Linux, que permite el uso de la cámara RGB y las funciones de profundidad.

Descarga en:
http://git.marcansoft.com/?p=libfreenect.git;a=commit;h=765 5fcf7239ba4907654089dba535a196685dbe5

2.2 PlayStation Move

PlayStation Move es un sistema de control de videojuegos mediante sensores de movimiento y detección de posición para la consola PlayStation 3 de Sony . Este sistema debe ser utilizado junto con el PlayStation Eye, ya que será el responsable de detectar la posición de los mandos.

2.2.1 Componentes

El mando “move”:

-Giroscopios (para establecer el ángulo en el que está posicionado el mando).

-Acelerómetros (miden la velocidad de reacción).

-Sensor de campo magnético (mide el viraje horizontal).

-Esfera que se ilumina (para poder situarnos en la habitación).

-Botones clásicos de PlayStaion.

-Conexión Bluetooth 2.0.

-Conexión USB 2.0.

Control de navegación:

-Botones clásicos de PlayStation

-Joystick analógico

-Conexión Bluetooth 2.0.

-Conexión USB 2.0.

PlayStation Eye:

-Micrófono con capacidad para reducir el ruido de fondo y prestar mayor atención a la voz. La entrada de audio tiene 4 canales: 16 bits por canal.

-Cámara RGB (120 frames por segundo a una resolución de 320×240 y 60 frames por segundo a 640×480).

2.2.2 Funcionamiento

El funcionamiento del PlayStation Move se asemeja en cierta medida al del Wiimote de Nintendo, que será explicado posteriormente. A grandes rasgos, al igual que el periférico de Nintendo, cuenta con acelerómetros para controlar la inclinación y posibles movimientos que podamos efectuar con el mismo, por otra parte, el giroscopio le proporciona a Move una precisión añadida, que detecta no solo los movimientos, sino también las rotaciones de éste, con una gran precisión.

A diferencia del Wiimote, en la parte superior del controlador, contamos con una bola luminosa. Esta bola, es la encargada junto con "eye", de posicionar el mando en el entorno de juego. De manera que, combinando tanto la información de los acelerómetros y giroscopio, junto con el posicionamiento preciso de la cámara, la PlayStation es capaz de saber en todo momento la posición exacta del mando, junto con su grado de inclinación,

reconociendo además, si acaba de ejecutar un giro sobre su eje, válido para juegos en los que los golpes con efecto (como puede ser el ping-pong) puedan modificar el resultado.

2.3 Microsoft Surface

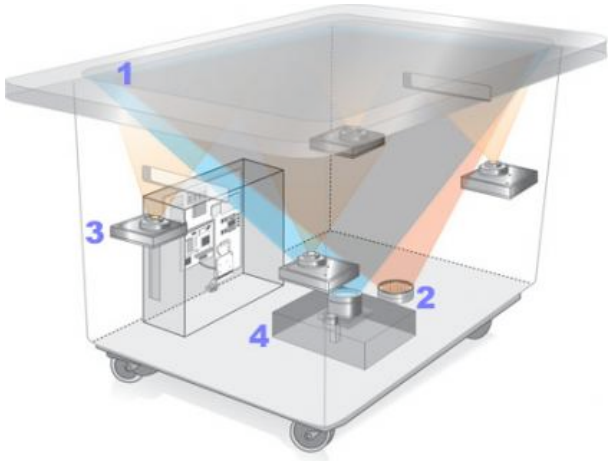
Microsoft Surface fue el primer ordenador multitáctil presentado por la compañía de Bill Gates en 2007.

Este prototipo no precisaba de teclado ni ratón, y gracias a la tecnología multitáctil podía ser utilizada por varias personas simultáneamente.

2.3.1 Primera generación

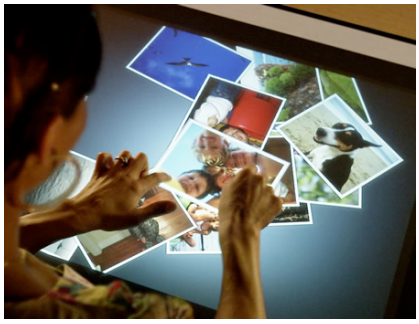
El prototipo constaba de:

- Pantalla de 30 pulgadas (76 cm.) empotrada en una mesa de dimensiones de 56 cm. de alto, 53 cm. de profundidad y 108 cm. de ancho. La parte superior de Surface era de acrílico y su marco interior estaba revestido de polvo de acero.
- El software corría bajo una versión especial de Windows Vista .
- Conexiones Ethernet 10/100, wireless 802.11 b/g y Bluetooth 2.0.
- Infrarrojos para reconocer objetos.
- CPU: Core 2 Duo, 2GB de RAM y tarjeta gráfica de 256 MB.
- Proyector.



Cuando este prototipo fue presentado, mostraba varias novedades. Entre ellas se encontraban:

- Era de los primeros elementos que permitía la no utilización de teclado y ratón para interactuar con el sistema, debido a que admitía actuar sobre este directamente con las manos, siendo capaz de reconocer varios puntos de contacto.



- Al contar con una pantalla horizontal, permitía una mayor interacción entre varios usuarios.

- Además de contemplar el tacto con las manos, era capaz de reconocer objetos situados sobre ésta, tales como: teléfonos móviles, cámaras de fotos o reproductores de música entre otras cosas, e interactuar con éstos, siendo capaz de intercambiar datos entre los objetos reconocidos, de una manera sencilla e intuitiva.

Para que esto funcionase correctamente, Surface contaba con varias cámaras debajo del cristal, las cuales eran las encargadas de mandar las imágenes que debían ser procesadas en el ordenador que incluía. Por otro lado, contaba con un proyector que proyectaba la interface sobre la parte inferior del cristal del surface.

Este prototipo apenas tubo repercusión comercial debido a su alto precio, cerca de 12500$ por mesa.

2.3.2 Segunda generación

En la actualidad, Microsoft ha presentado su segunda versión, la cual a sido implementada por Samsumg, "Sur40". Tecnológicamente, esta versión cuenta con:

- Procesador AMD Athlon II X2 de 2.9GHz ,GPU Radeon HD 6700M.
- Superficie de 40" en lugar de 30" con cristal Gorilla Glass y Full HD, siendo capaz de reconocer 50 simultáneamente.
- Windows 7.
- Tecnología PixelSense.

Con la utilización de la tecnología PixelSense, se desprende del proyector y las cámaras, pudiendo hacer que la mesa ocupe un grosor mucho más reducido. Esto se debe a que en esta versión, cada píxel es un sensor infrarrojo, siendo capaz de reconocer hasta 50 puntos diferentes simultáneamente.

Además, los costes de la versión "Sur40", se han reducido a unos 7500$, siendo el Banco Real de Canadá uno de los primeros en adquirir estos equipos, ya que claramente, debido a su precio, estas mesas no están orientadas al uso personal sino al comercial.

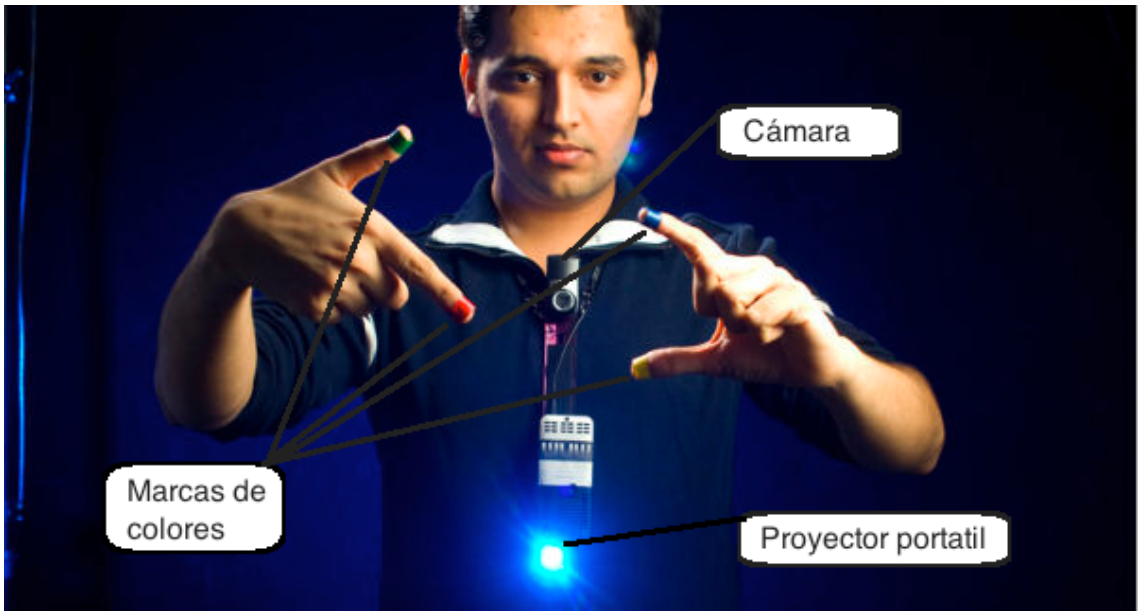
2.4 Interfaces naturales “6TH SENSE”

Desarrollado por el Instituto Tecnológico de Massachusetts (MIT) , “6th sense” es un interface gestual portable, que aumenta el mundo físico que nos rodea añadiéndole información digital, y que nos permite utilizar los gestos naturales de las manos para interactuar con esa información.

Este gadget tiene la particularidad de estar pensado para poder ser construido con componentes que podemos encontrar en cualquier tienda, de hecho, el prototipo costó alrededor de 350$. De esta manera los componentes del prototipo son: un proyector portátil, una cámara y un teléfono móvil.

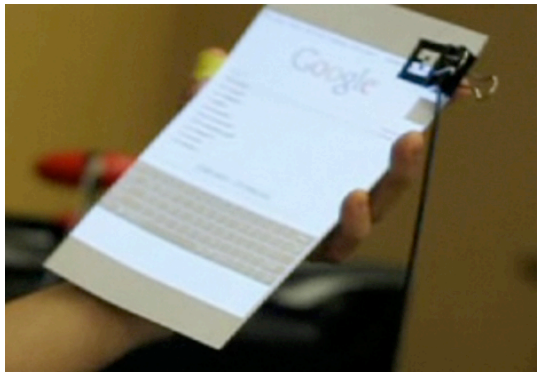
Por otro lado, para que el programa detecte los gestos, se deben colocar unas marcas de colores en las puntas de los dedos.

El equipo quedaría así:



2.4.1 Funcionamiento

El proyector portátil es el encargado de proyectar la información visual en cualquier tipo de superficie (paredes, mesas, en nuestras propias manos), haciendo así que estas se conviertan en nuestra interface.



A su vez, la cámara es la encargada de suministrarle al teléfono las imágenes que tendrá que procesar para interpretar así los movimientos y gestos de nuestras manos y dedos. Estos gestos actuaran de instrucciones de interacción con la aplicación proyectada.

Es por esto que con simples gestos sixthsense, es capaz de lanzar la aplicación de "teléfono" proyectándola sobre nuestras manos, reconocer un periódico y proyectar un video sobre este o, lanzar la cámara de fotos para sacar una foto.

3.-WIIMOTE

3.1 Introducción

El Wii Remote o Wiimote, es el mando de control de la videoconsola Wii, lanzada el 19 de Noviembre de 2006 por la compañía japonesa Nintendo y desarrollada junto con la colaboración de IBM y ATI.

El objetivo de la compañía, al lanzar ésta nueva consola, no fue la de mejorar únicamente el rendimiento de la anterior, si no que intentaron innovar la forma de juego, la interactividad jugador-videoconsola, y ahí es donde reside su éxito. Para ello desarrollaron el mando Wii Remote que, combinando una serie de acelerómetros incorporados en este, botones clásicos de mandos, ambos comunicándose mediante bluetooth, y una cámara infrarroja incorporada en el mando, dan una experiencia de juego e interacción no conocida en el mundo de las videoconsolas hasta entonces.

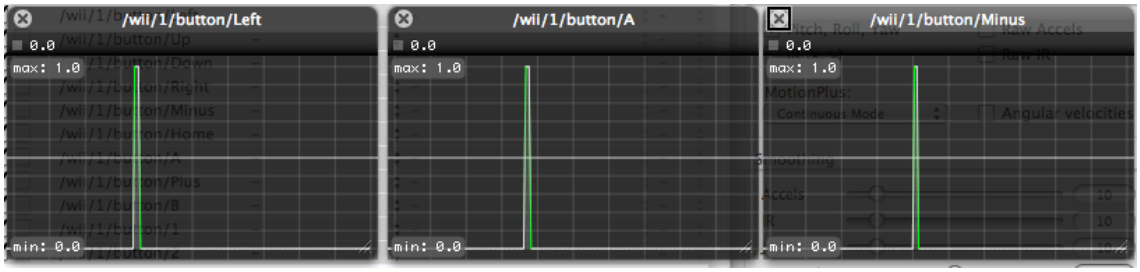
3.2 Características

El Wiimote cuenta con:

- Botones (todos ellos digitales)
- Altavoz
- Acelerómetro
- Leds
- Motor vibratorio “Rumble”
- Bluetooth
- Conexión para periféricos

3.2.1 Botones

El mando cuenta con 7 botones (Minus, plus,home, power, 1, 2, A) uno de los cuales colocado en la parte posterior (B), y con una cruz (entendida en la trasmisión de datos como 4 botones Left, Rigth, Up, Down).

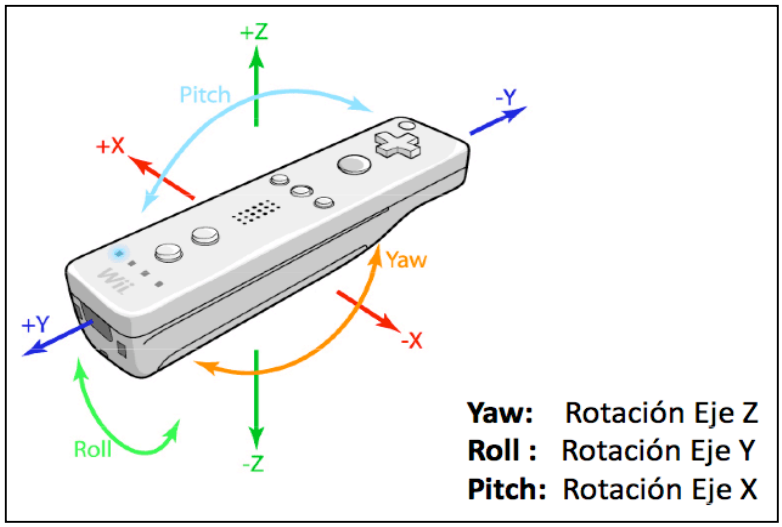


Como podemos ver en la imagen, se han puesto como ejemplo los botones: Left (de la cruz), A y Minus. Observamos que todos ellos son señales digitales que varían de 0 a 1.

3.2.2 Altavoz

El dispositivo cuenta con un altavoz situado en la parte central del mando, utilizado para transmitir un mayor realismo en el juego, simulando el movimiento de una espada, sonido de disparos etc.

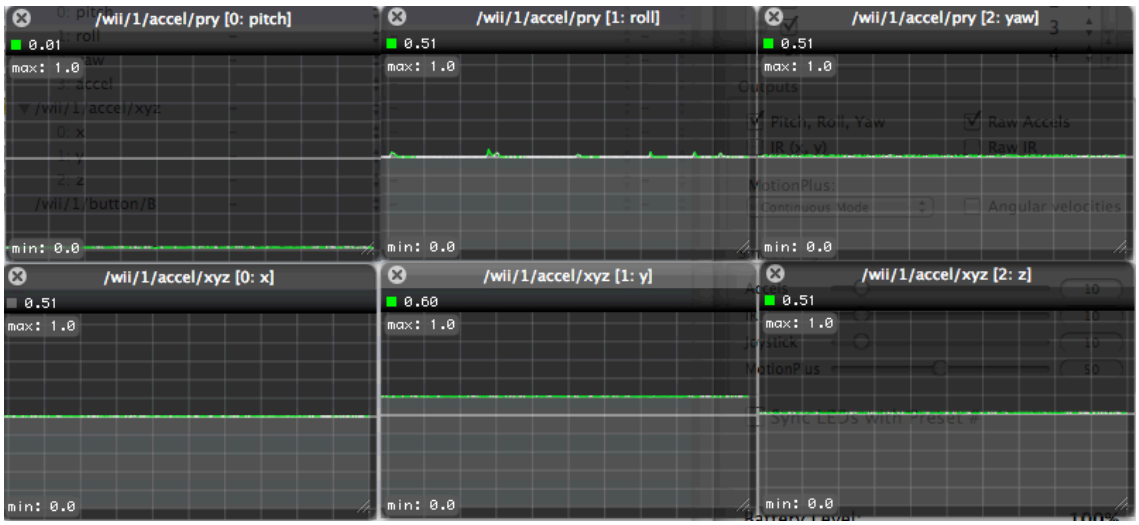
3.2.3 Acelerómetro



Un acelerómetro es un instrumento para medir aceleración, detectar y medir vibraciones, o medir aceleración debida a la gravedad (inclinación).

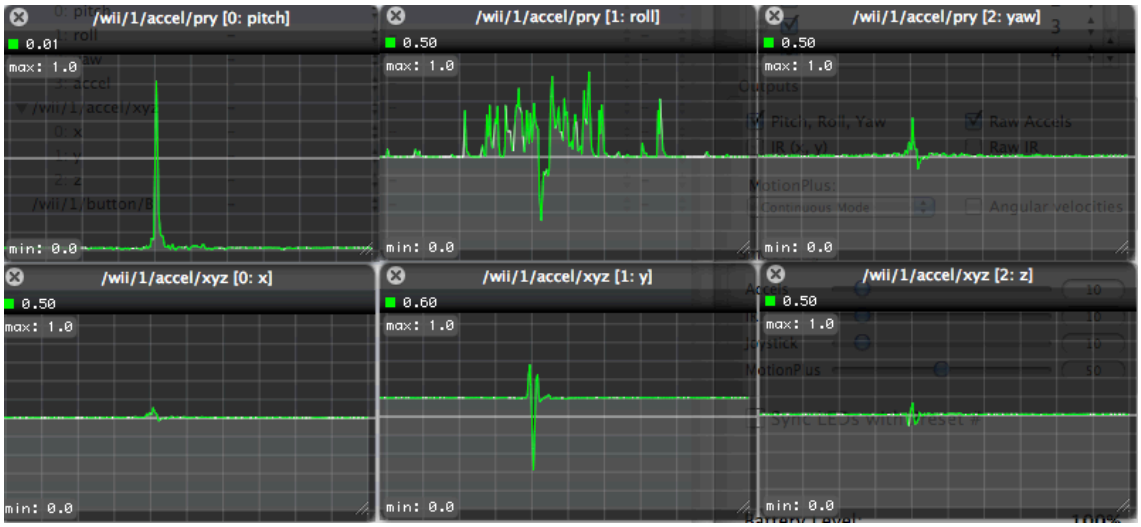
Las características principales de éste acelerómetro son su reducido tamaño y su bajo consumo, todo ello para hacer que las baterías duren más.

Concretamente cuenta con el chip ADLXL330 de la compañía Alalog Devices, que le proporciona medidas de rotación: Yaw, roll y pitch en los tres ejes, además de aceleración en los ejes X, Y, Z. La diferencia entre estas señales es que yaw, roll y pitch nos mide la rotación en el eje que tiene el wiimote, como podemos observar en la figura. Por ejemplo si rotamos 90º en el eje X lo dejamos quieto en esta posición (poner el mando en posición vertical), la señal proporcionada por el Wiimote será esta:



El pitch de rotación nos daría valor 0, por otro lado rol y yaw 0.5 (el valor de la posición neutra es 0.5), por el contrario, los valores de rotación serían estables, pese a que se detecte que en el eje Y existe una variación de 0.1.

De esta manera, si mantenemos el mando en la misma posición y lo agitamos en este mismo eje (Y) (en posición vertical, sacudirlo de arriba abajo), la señal nos mostraría esto:



Descartando las interferencias en roll debidas al movimiento no perfecto, observamos que la sacudida se ve reflejada en la rotación pitch (siendo la sacudida interpretada como una rotación muy rápida), y una aceleración más notable en el eje Y. Así, con estas seis señales, se pueden detectar tanto la posición del mando como la aceleración ejercida en cualquiera de los tres ejes.

3.2.4 Leds

Los led están colocados en la parte inferior del mando, indicando el número de mando que está conectado a la consola (máximo de 4), e indicando, cuando parpadean todos a la vez, que se esta intentando establecer conexión con la Wii.

3.2.5 Motor vibratorio “Rumble”

Este motor vibratorio o “Rumble”, está situado en la parte interna de Wiimote, siendo el encargado de aportar más realismo en los juegos. Este tipo de elementos se conocen como tecnologías “hápticas” (interfaces tecnológicos que interaccionan con el ser humano mediante sensaciones no visuales o auditivas).

3.2.6 Bluetooth

Bluetooth, ha sido la tecnología encargada de la comunicación entre Wiimote y Wii. Esta tecnología permite que los mandos sean reconocidos a una distancia de 10 m de la consola, con un máximo de 4 mandos.

3.2.7 Conexión para periféricos

La conexión para periféricos se encuentra en la parte inferior del mando. En ésta se pueden conectar el Nunchuck, añadiéndole varios botones de control más, un joystick analógico, un acelerador, y por otro lado el Wiimotionplus que ayuda al Wiimote a aumentar la sensibilidad de movimientos de éste con un nuevo sensor.

3.3 Comunicación

El protocolo elegido por Nintendo para la comunicación entre dispositivos es el OSC (Open Sound Control), que ha ganado puestos frente al midi gracias a su potencia y su flexibilidad. Al igual que el midi, no sólo es utilizado para el control de audio, si no que también pude ser empleado para el control de periféricos, conexión entre dispositivos, etc.

3.3.1 OSCulator

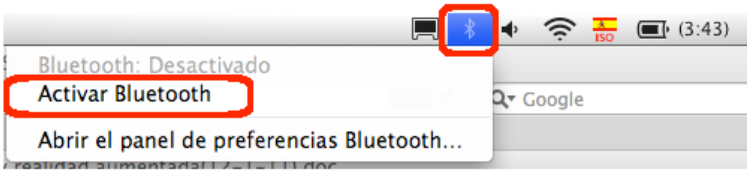
En este apartado, explicaremos como llevar a cabo la conexión entre Wiimote y un ordenador Mac, concretamente con un Macbook (white). Para ello utilizaremos el programa OSCulator, desarrollado por Camille Troillard en Design & Programming que podremos descargarlo desde la página http://www.osculator.net/. Al ser una versión gratuita, cada 5 minutos nos recuerda que podemos registrarnos en su página web. La potencia de este programa está en el hecho de ser capaz de recibir datos en protocolo OSC y convertirlos a multitud de protocolos.

El ordenador con el que contamos es un Macbook, de manera que lleva bluetooth incorporado, de no ser así deberíamos adquirir un dispositivo bluetooth y conectarlo.

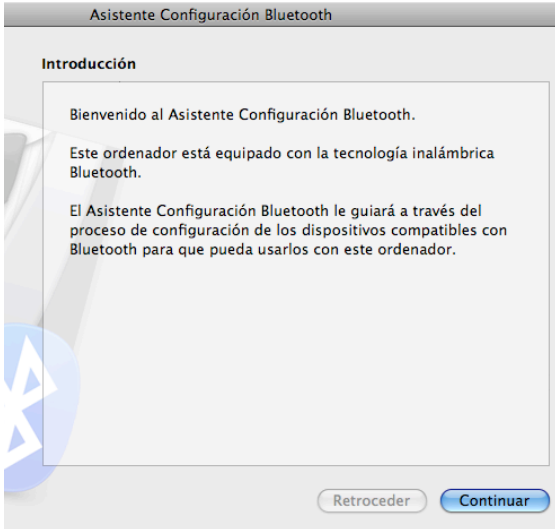
Pasos a seguir para conectar Wiimote a nuestro Macbook.

Paso 1:

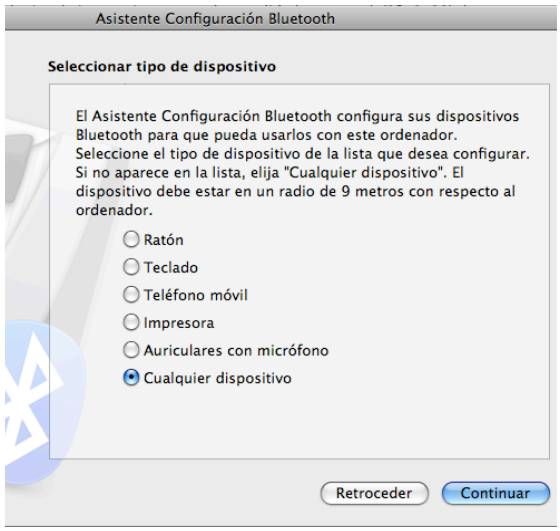
Inicialmente deberemos activar el bluetooth de nuestro ordenador. Para ello, por defecto, deberemos ir a la parte superior derecha del escritorio y buscar el icono bluetooth indicado en la imagen, presionarlo y elegir “*Activar bluetooth”*.



Una vez activo, volveremos a presionar sobre el icono bluetooth y el menú habrá cambiado, teniendo que presionar esta vez sobre "*Configurar dispositivo bluetooth*". Una vez hecho esto, nos aparecerá la siguiente ventana (imagen 1), donde tras pulsar "*Continuar",* nos mostrará otra ventana (imagen 2).



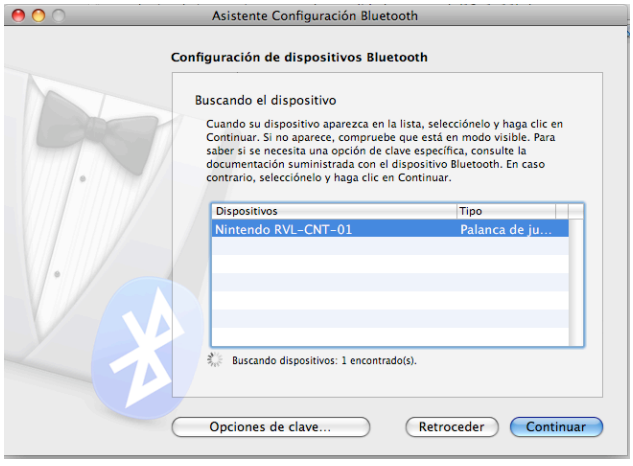
(Imagen 1)



(Imagen 2)

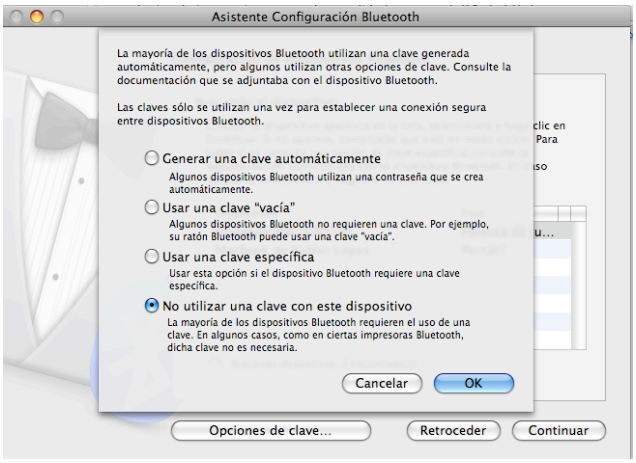
En esta ventana, seleccionaremos "*Cualquier dispositivo"* y presionaremos "*Continuar*".

Una vez llegados a este punto, el bluetooth del ordenador comenzará a buscar un dispositivo al cual asociarse, de manera que tendremos que activar nuestro Wiimote. Para ello, sólo tendremos que presionar los botones del mando, 1 y 2 simultáneamente. Si lo hemos hecho correctamente, los leds de la parte inferior de Wiimote parpadearán a la vez. De ésta manera, y si hemos seguido los pasos anteriores correctamente, el ordenador ha tenido que ser capaz de encontrar nuestro dispositivo y mostrarnos algo parecido a esto:



Se abrirá una ventana que nos mostrará una lista con todos los dispositivos blueetooth que ha encontrado, entre ellos nuestro Nintendo Wii Remote. Si no lo ha encontrado, repasamos los pasos anteriores.

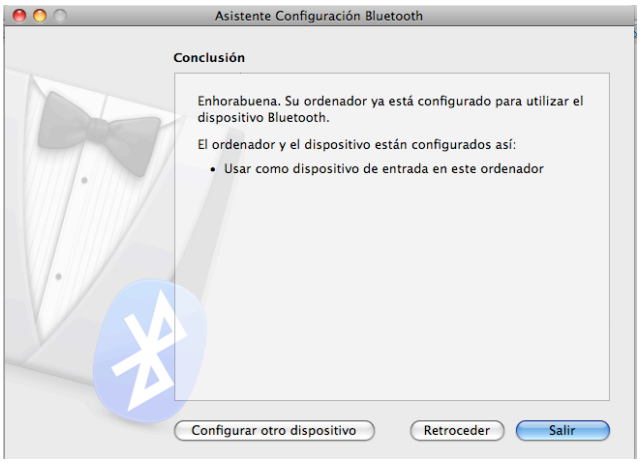
Una vez encontrado el dispositivo, pulsaremos sobre el botón "*Opciones de clave"* y le diremos que no use ningún tipo de clave con este dispositivo:



Tras lo cual, presionaremos "*OK*" y "*Continuar"*. Hecho esto, se abrirán una serie de ventanas (imágenes 3,4 y 5) que tendremos que confirmar.

(Imagen 3)

(Imagen 4)

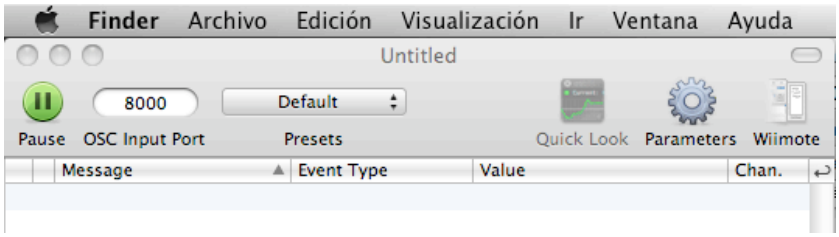


(Imagen 5)

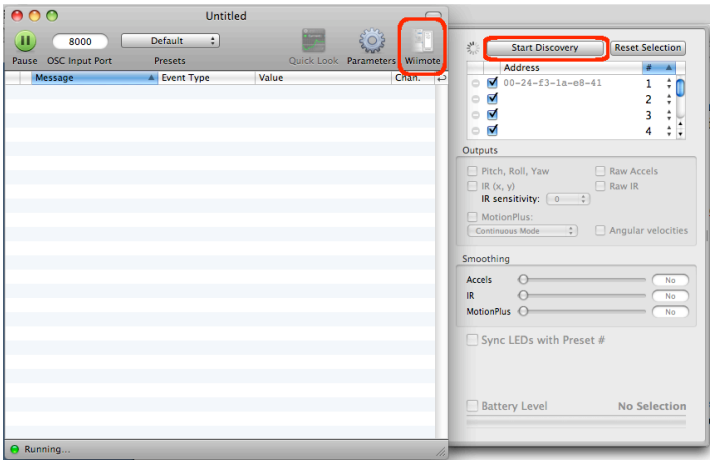
Puede darse el caso de que, si nos demoramos demasiado, en algún momento el ordenador deje de detectar el dispositivo. Si esto llega a ocurrir, presionaremos nuevamente los botones 1 y 2 del Wiimote.

Paso 2:

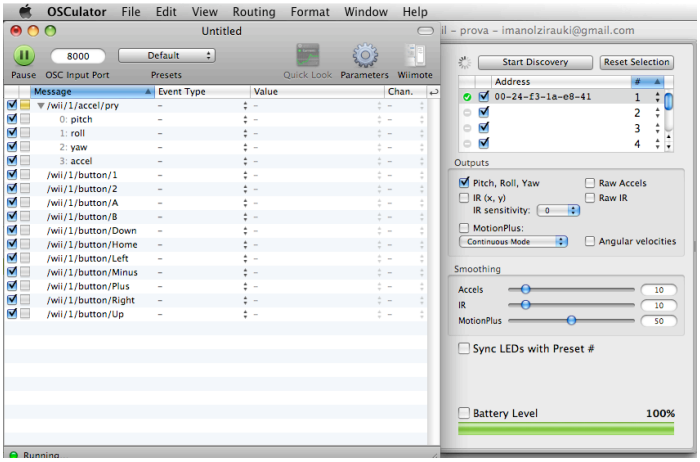
Una vez realizados los pasos anteriores, es hora de lanzar el programa OSCulator. Este, debería de presentar el siguiente aspecto.



A continuación, presionaremos el botón/icono que está situado en la parte superior derecha con el icono de Wiimote, tras lo que se desplegaría una ventana, dejándolo con este aspecto.



Llegados a este punto, solo nos quedaría presionar el botón "*Start Discovery*" a la vez que pulsamos nuevamente los botones 1 y 2 del Wiimote.



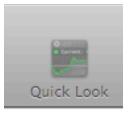
Automáticamente el dispositivo debería ser reconocido por el programa, mostrándonos los parámetros que reconoce. Por otra parte, sabremos que éste ha sido

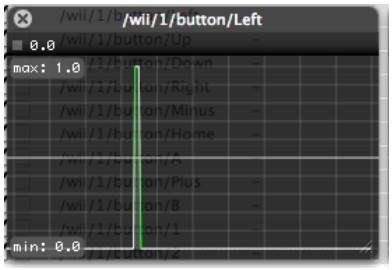
conforme, porque los leds dejarán de parpadear para que uno de ellos (normalmente el primero, señalado con un punto) muestre una luz fija.

En principio, solo reconocerá los datos transmitidos por el acelerómetro, pero si vamos pulsando uno por uno los botones del mando, veremos como éstos se añaden a la lista. Por defecto, nos aparecerá marcada la opción de "*Pitch, Roll, Yaw"* en la parte derecha de la ventana, siendo éstos los parámetros del acelerómetro que reconoce, pudiendo ampliar dichos parámetros marcando las demás opciones.

Paso 3:

Una vez llegados a este punto, nos queda la configuración del programa, tendiendo que indicar a qué formato queremos que nos traduzca los datos recibidos en protocolo OSC. Para ello es recomendable conocer el tipo de señal que recibe éste. Pudiendo ser de tipo pulso digital (en el caso de los botones) o señal analógica (acelerómetro). Para ello podemos pulsar sobre el botón "*Quick Look*" , éste nos mostrará un gráfico de la señal que está recibiendo en ese momento, oscilando sus valores entre cero y uno.





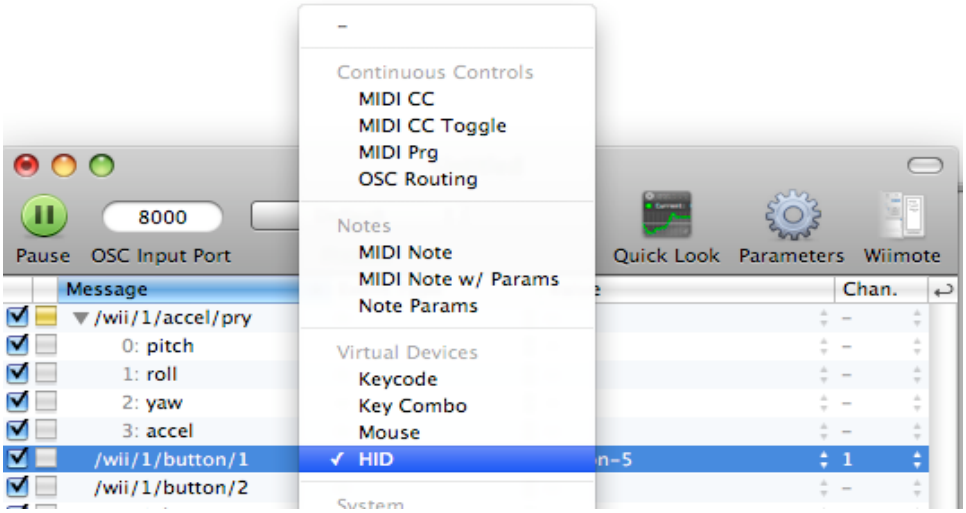
3.4 Blender

Blender es un programa de modelado 3D y animación, multiplataforma y de código libre, es decir, es capaz de ser instalado en la mayoría de los sistemas operativos actuales, existiendo versiones para: Windows, Mac OS, Linux, Solaris e Irix , contando además con la particularidad de ser de código abierto bajo la licencia GNU GPL.

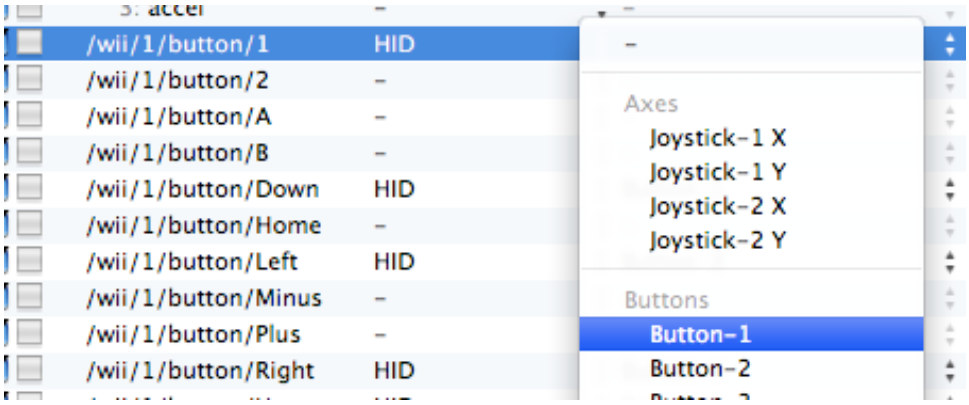
Para su instalación en el sistema operativo MAC OS, únicamente tendremos que acudir a la página web oficial de Blender y, en apartado de descargar, elegir la versión correspondiente a nuestro sistema operativo: http://www.blender.org/download/get-blender/ . Una vez elegida nuestra versión, nuestro equipo descarga un archivo comprimido .zip, el cual en nuestro caso, únicamente tendremos que descomprimirlo y ejecutar el archivo Blender.

3.4.1 Configuración OSCulator para su utilización en Blender

Para que el Wiimote interactúe con Blender tendremos que configurar adecuadamente OSCulator. Para ello, conectaremos éste al ordenador tal y como se indica en el apartado anterior. Una vez conectado, presionaremos encima de la línea que se muestra en la primera columna en la columna de "Event Type", con lo que se abrirá el siguiente desplegable.



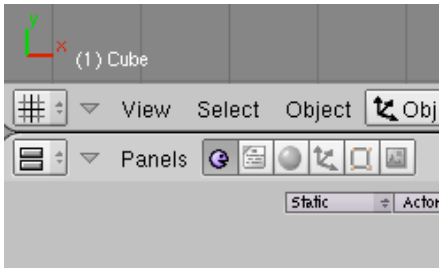
En este menú desplegable, seleccionaremos HID y haciendo click en el siguiente guión de la columna "Value", seleccionaremos el número de botón que queramos, en nuestro caso, seleccionaremos "Button-1" , haciendo lo mismo con los demás botones pero eligiendo "Button-2" y así sucesivamente.



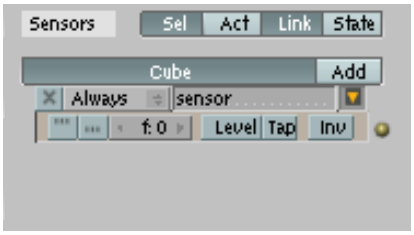
Una vez configurados todos los botones que queremos utilizar, pasaremos a la parte de Blender.

3.4.2 Configuración Blender

En este apartado veremos la forma en la que el mando Wiimote interactúa con el programa Blender.



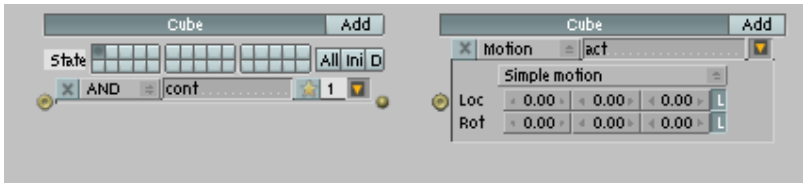
Para ello deberemos seleccionar el icono de la cara sonriente que aparece en la parte inferior izquierda de la pantalla.



Una vez presionado, se abrirá un menú como el de la imagen. A continuación, tendremos que presionar el botón "Add" que se muestra en la parte derecha de "Cube" para que nos aparezca un nuevo evento.

Nuevamente presionaremos el botón "Add" que

aparece en los consecutivos “Cube” que se muestran a la derecha, debiendo quedar una configuración similar a esta.



Llegados a este punto, debemos configurar Blender para que reciba las señales enviadas por el wiimote.

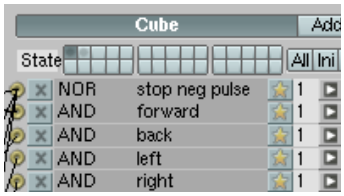
Para ello, desplegaremos el menú de movimientos del cubo, presionando encima donde pone “Always”, y seleccionaremos Joystick , ya que este será este el tipo de señal al que se traducirán los datos por parte de OSCulator.

El botón “Index”, nos indica el número del joystick al que vamos a asignar los movimientos, en nuestro caso, como tendremos un solo mando, se le asignará el índice 0. Tenemos que atender que en OSCulator nos da la opción de controlar dos mandos, pudiendo asignar como índice 1 o 2, por el contrario en Blender, los índices comienzan en 0. De esta manera los índices asignados en OSCulator siempre serán un valor superior al asignado en Blender.

Dicho esto, tendremos que considerar, de igual manera, la asignación de los botones en OSCulator, es decir, si al botón Up le asignamos “Button-1”, en Blender será el “Number”: 0.

Una vez aquí, con cuatro botones configurados, podríamos tener el control de la cruz del wiimote, es decir, control sobre: arriba, abajo, derecha e izquierda.

El siguiente paso, es asignar una acción a cada botón.

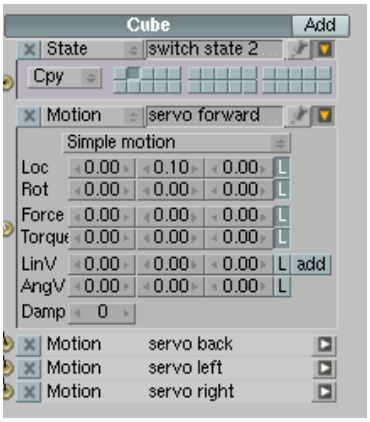


Para ello, pasaremos a la siguiente columna, y presionaremos el botón “Add” y en lugar de dejar “AND” como nos viene por defecto, seleccionaremos “NOR”, para que esta acción se active cuando ninguna de las otras lo esté.

Seguidamente añadiremos cuatro acciones más, a las cuales les podemos dar los nombres de la acción que van a ejecutar, tal y como muestra la imagen.

Finalmente, pasaremos a la siguiente columna. Aquí será donde asignaremos las acciones que deseemos a los diferentes botones.

En este apartado, igual que en los anteriores, presionaremos el botón “Add” para añadir una nueva acción.

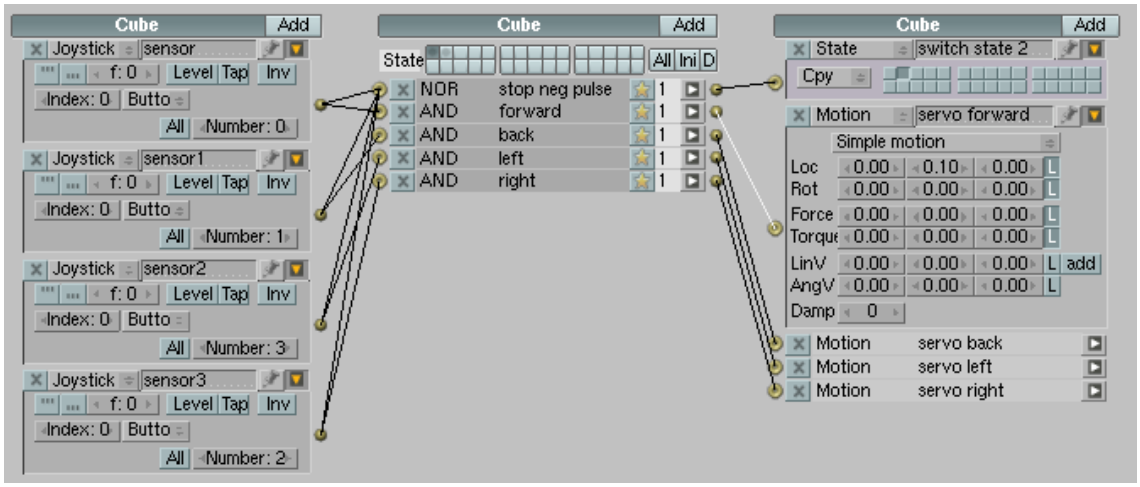


La primera acción que añadiremos, será la acción que debe ejecutarse cuando ninguno de los botones sea pulsado.

Para ello, presionaremos el botón "Add" pero en lugar de dejar la acción "Motion" que nos da por defecto, seleccionaremos "State".

Para las siguientes acciones, pincharemos nuevamente "Add", pero esta vez dejaremos por defecto "Motion". Para asignarle movimiento a nuestro objeto, nos fijaremos en las tres columnas y siete filas de números que aparecen. En nuestro caso, solo queremos asignarle movimiento en dos de los ejes, por lo que centraremos nuestra atención en la primera fila. De izquierda a derecha, controlamos el movimiento en el eje Y (izquierda - derecha), el eje X (adelante - atrás) y el eje Z (arriba – abajo). Dependiendo del tamaño del ejemplo que hayamos creado, y de la velocidad que le queramos asignar, pondremos unos valores u otros, en nuestro caso los valores han sido de 0,1 y -0,1.

En este momento, solo nos queda unir botones con acciones. Para ello, si observamos en la parte derecha de cada botón tenemos un circulo de color dorado, pincharemos en él y arrastraremos hasta la acción que queremos que realice.



Deberemos fijarnos que todos los botones están unidos con la acción "NOR" y luego, cada uno independientemente, con su acción particular. Si todo se ha hecho correctamente debería quedarnos con un aspecto como este.

Para comprobar que todo esta configurado adecuadamente, nos aseguraremos de que el OSCulator está activado y a continuación presionaremos la tecla del teclado "p" para ejecutar nuestra prueba. Si todo está correcto, al presionar los botones arriba, abajo, izquierda y derecha, nuestro objeto debería moverse.

3.5 Wiiflash

Otra de las posibilidades que nos aporta el mando de la Wii, es la interacción con flash. Existe una plataforma llamada Wiiflash desarrollada por Joa Ebert y Thibault Imbert sin ánimo de lucro, la cual es compatible tanto para Mac como para Windows.

Este proyecto fue desarrollado con la intención de hacer posible la interacción del Wiimote con aplicaciones que utilizasen flash.

Actualmente, a pesar de estar en fase beta (de desarrollo) desde 2008 es compatible con las librerías de Papervision con lo que hacen posible el manejo de objetos en 3D con las mismas.

Esta aplicación puede descargarse desde su página web oficial: http://wiiflash.bytearray.org/ en el apartado del "Downloads" en el cual podremos encontrar un paquete donde incluirá el soporte para ambas plataformas (Windows y Mac) dentro de una carpeta llamada "Servers".

Para su utilización sobre la Plataforma Mac, tendremos que tener en cuenta que esta es una aplicación de 32bit y en aquellos equipos que funcionen a 64 bits deberán ejecutarla como tal, por lo tanto, simplemente tendremos que abrir las opciones de archivo "WiiFlashServerJ" que se encuentra dentro de la carpeta "Servers", o bien haciendo botón derecho sobre este y seleccionando "Obtener información", o bien pulsando la combinación de teclas "cmd + i". Una vez hecho esto se nos abrirá un cuadro como el de la izquierda.

Como vemos en la imagen tendremos que seleccionar la opción "Abrir en modo de 32 bits".

A continuación, los pasos a seguir serán:

1- Activar el bluetooth
2- Ejecutar el programa Wiiflash
3- Pulsar los botones 1 y 2 del Wiimote para que este se sincronice con Wiiflash
4- Ejecutar cualquiera de las aplicaciones que vienen en modo de prueba dentro del programa en la carpeta "Examples".

Los inconvenientes de esta plataforma son, que puesto que se ha quedado en una versión beta, su funcionamiento no es del todo estable, sobre todo sobre la plataforma Mac. Al ser esta una tecnología que no está desarrollada completamente, no existen especificaciones claras de como hacer funcionar correctamente la aplicación, y son muchas las pruebas que hay que realizar para un mínimo funcionamiento.

Por otra parte una vez en funcionamiento, la estabilidad dura un breve período de tiempo, puesto que parece ser que por algún tipo de problema utiliza todos los recursos de la memoria RAM y al minuto de ponerse en funcionamiento, este se vuelve poco estable y hace que la aplicación se bloquee, teniendo que forzar el reinicio de esta (para forzar el reinicio de una aplicación sobre la plataforma Mac, con la tecla “alt” presionada, haremos botón derecho sobre el icono de la aplicación, de esta manera en lugar de la opción de salir, nos dará opción a forzar salida).

4.- REALIDAD AUMENTADA

Llamamos realidad aumentada cuando añadimos datos o información virtual a una visión directa o indirecta de un entorno físico del mundo real, en tiempo real. La gran diferencia entre realidad virtual y realidad aumentada, data en que la realidad virtual sustituye a la realidad, mientras que la realidad aumentada nos añade información en tiempo real sobre el mundo real.

Actualmente son muchos los caminos abiertos en lo que la realidad aumentada se refiere, pero en este caso nos vamos a centrar en la realidad aumentad sobre flash combinada con AcctionScript librerías de openGL, la cual esta en continuo desarrollo y parte con la ventaja de que en la red se puede encontrar una gran cantidad de ejemplos con código abierto, descargables y manipulables por el usuario, para ello contaremos con la herramienta Flartoolkit, basada en Artoolkit, pero orientada a flash.

4.1 Artoolkit y Flartoolkit

Artoolkit es una API (Application Programming Interface) para el desarrollo de aplicaciones de realidad aumentada. Sus características principales son:

1.- Tiene una licencia GPL (General Public License), o lo que es lo mismo, que su código es libre y se puede manipular como uno quiera, siempre que no se obtenga ningún beneficio económico con este.
2.- Se considera un estándar en el campo de la realidad aumentada.
3.- Es multiplataforma.
4.- Es muy rápido.

En nuestro caso hemos optado por utilizar Flartoolkit, que como antes comentábamos, está basado en la tecnología de Artoolkit pero adaptada a Flash. Se ha optado por la tecnología flash debido a su fácil utilización vía web, puesto que tenemos que pensar los posibles usos que tenga esta tecnología, Internet es un campo muy atractivo para darle salida.

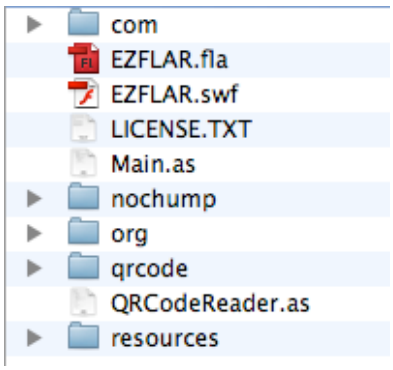
4.2 EZFlar

Concretamente el código utilizado a sido desarrollado por EZFlar y a sido descargado de su página web oficial en la dirección : http://www.ezflar.com/home/show_download. En esta página además de darnos la opción de descargarnos el código, podemos hacer una aplicación de realidad aumentada online, sin necesidad de manipular ningún código, directamente con las herramientas que EZFlar nos pone a nuestra disposición.

Este código, desde un inicio nos permite trabajar con una gran cantidad de formatos, concretamente:

Para ello solo tenemos que tener en cuenta la organización de carpetas que tiene el programa, pero eso lo explicaremos más adelante.

Una vez descargado el código nos encontramos con la carpeta "tcha-tcho-EZFLAR-cab94f2", en cuyo interior hallaremos un archivo "README.textile", en el que localizaremos una pequeña explicación del código utilizado, y ciertas recomendaciones para una ampliación y mejora de este, y por otro lado una carpeta "src" que contiene todas las carpetas y ficheros. En su interior, podemos ver esta organización:



Las más importantes serán la carpeta "org", que contendrá las librerías que necesita nuestro programa, entre ellas: la carpeta "ascollada", que contiene las librerías para el manejo de archivos en 3d ".dae". La carpeta "libspark", que contiene a la carpeta "Flartoolkit", con todas las librerías que le hacen falta para el manejo de la realidad aumentada. Y por último la carpeta "Papervision3d" que contiene las librerías para el manejo de gráficos en 3d sobre Flash.

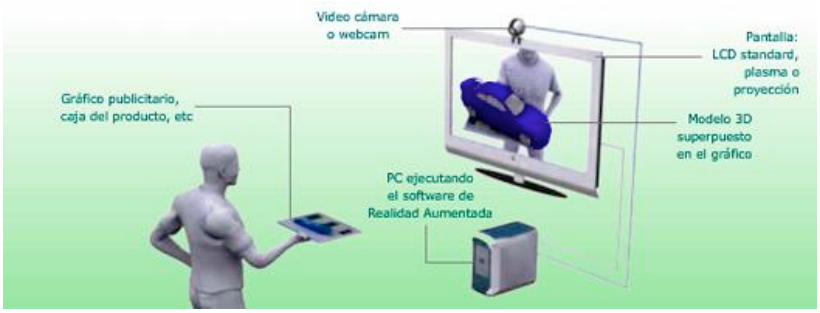
Por otro lado, otra carpeta que nos interesará será "resources" que por una parte contendrá los patrones de búsqueda .pat en la ruta "src/resources/flar/patterns", de la que más adelante explicaremos su funcionamiento. Y por otra parte "src/resources/models", que contendrá todos aquellos archivos de audio-3d-imagenes-video que queramos que sean asociados a los patrones .pat.

Y por último, la carpeta "com", en la cual hallaremos todos los archivos que controlan los diferentes formatos que podemos desarrollar con nuestra aplicación de realidad aumentada en la ruta "src/com/tchatcho/constructors", y sobre los cuales podremos modificar ciertos parámetros específicos de cada formato.

Por otra parte, además de las carpetas antes mencionadas, nos encontramos con los archivos principales de control de la aplicación, como puede ser "QRCodeReader.as" que controla los parámetros de la cámara y Main.as que será el archivo principal sobre el que trabajaremos, y que permitirá cambiar tanto los patrones que queremos asociar como los objetos asociados a estos.

4.3 Funcionamiento

La idea principal de la organización de nuestra aplicación de realidad aumentada será esta:

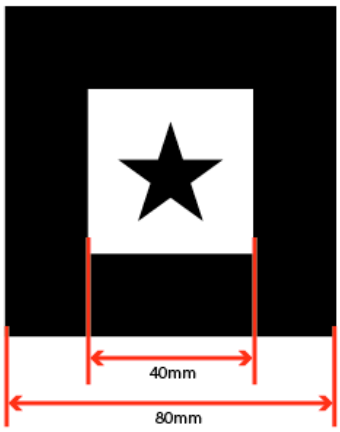


Partiendo de este esquema comenzaremos a trabajar con nuestro proyecto, y lo primero que tenemos que hacer es generarnos un marker.

Estos marker o patrones podrán crearse con cualquier programa de dibujo, siempre y cuando respetemos las características que vamos a comentar a continuación.

4.3.1 Markers

El programa inicial entenderá que nuestro marker será un cuadrado negro de 80mm de altura, en cuyo interior y centrado se encontrara otro cuadrado de 40mm, blanco en este caso. Será en el interior de este donde colocaremos el dibujo el que queramos que nuestra cámara detecte. Como observamos en el ejemplo, el dibujo deberá ser en blanco y negro, ya que nuestro programa hace una umbralización del dibujo y lo transforma a blanco y negro. De esta manera, si nuestro dibujo es en blanco y negro también, facilitaremos que el programa reconozca nuestro patrón y así funcione correctamente.

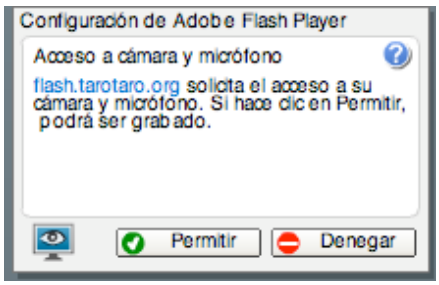


Podríamos hacer un patrón con diferentes medidas, pero esto daría lugar a que el programa interpretase que tiene una inclinación y se encontrara a una distancia no correspondiente con la realidad.

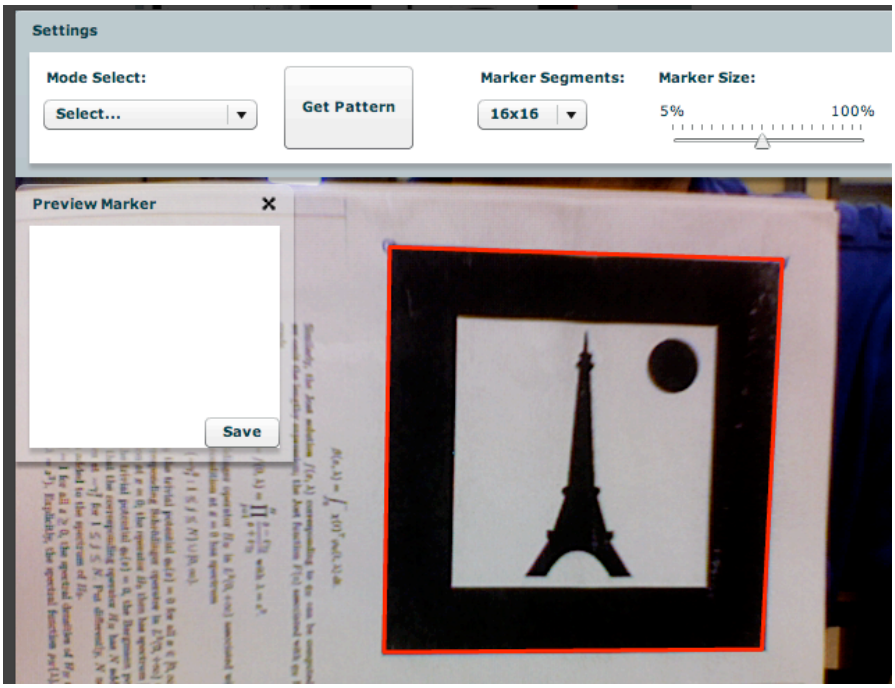
Otro de los aspectos a tener en cuenta a la hora de hacer nuestro patrón, son las características del dibujo central. Existen varias formas de facilitar el trabajo a nuestro programa para que encuentre y calcule la posición exacta de nuestro marker. La primera de ellas, es no hacer dibujos excesivamente complejos, puesto que a este le resultaría costoso encontrar y definir bien el marcador. Otra forma, es añadirle un circulo negro en una de las esquinas del cuadro, ya que así le ayudaremos a encontrar rápidamente la orientación de este.

Una vez tenemos el dibujo finalizado, tenemos que digitalizarlo en un formato que nuestro programa entienda. Este es el formato .pat. Para ello existen varias formas de hacerlo, una de ellas es accediendo a la página web http://flash.tarotaro.org/blog/2008/12/14/artoolkit-marker-generator-online-released/ y pinchando sobre el link “ARToolKit marker Generator Online” que se encuentra en el paso 2 de cómo se genera un marker. De esta forma tenemos la opción de imprimir nuestro dibujo y acudir a la dirección antes mencionada o directamente acceder a la dirección con el dibujo en formato .jpg.

Para que esto funcione tendremos que tener instalado el Plugin 10.0 de Flash. Si todo está instalado correctamente, se abrirá una ventana que nos pedirá permiso para la utilización de la webcam.



Presionaremos “Permitir” y aparecerá una ventana como esta:

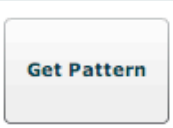


En esta ventana existen varios parámetros que podemos ajustar a nuestras necesidades, como puede ser el tamaño del dibujo “Marker Size”, el número de segmentos con que queremos que cuente nuestro marker (a mayor numero de segmentos nuestro dibujo en formato .pat tendrá más precisión, pero a su vez será más pesado, en lo que a volumen de datos se refiere, y podría hacer que el funcionamiento se viera ralentizado) ”Marker Segments” y por último el modo en que queremos que nuestro marker sea detectado “Mode select”. En este último parámetro podemos obtener nuestro marker de dos manera diferentes:

O bien, cogeremos la copia impresa de nuestro marker y se la mostraremos a la cámara hasta que el cuadro negro quede rodeado de una línea roja. Un parámetro que tenemos que tener muy en cuenta es la luz ambiental que tenemos a la hora de generar nuestro marker, ya que puede darse el caso de que el programa sea incapaz de detectar el patrón debido a la poca luz ambiental. (Este será un campo que tendrá que ser muy tenido en cuenta cuando nuestro proyecto de realidad aumentada empiece a funcionar).

O desplegamos este último parámetro y seleccionamos “load marker image”, lo cual nos dará opción a seleccionar un archivo de nuestro ordenador que contenga el marker, por lo que la precisión de este será mayor, evitándonos el margen de error que pueda añadirle la webcam.

Una vez nuestro marker sea detectado, tanto para la opción de la webcam como para la selección manual del archivo, presionaremos el botón "get pattern", lo que nos mostrará una imagen previa, para que valoremos si la imagen capturada se asemeja a la original, de no ser así, jugaremos con los parámetros antes mencionados (como por ejemplo poner al 100% el valor del "Marker size" o aumentar el numero de segmentos de nuestro .pat).



Cuando consideremos que la imagen es correcta, presionaremos el botón "save", y guardaremos el .pat, con el nombre que veamos conveniente, auque más adelante veremos cuales son los nombres que tendremos que darle si queremos trabajar con varios marcadores a la vez.

Para finalizar, copiaremos los marker que hagamos en la carpeta "src/resources/flar/patterns".

4.3.2 Objeto

Como se explicaba en la primera parte, los objetos que queremos que salgan representados en nuestra aplicación de realidad aumentada pueden ser de varios tipos: Audio, video, imagen, 3D, url.

El primer paso que tendremos que dar para poder utilizarlos será elegir el objeto que queremos que se muestre cuando la cámara detecte nuestro marker y copiarlo en la carpeta "src/resources/models".

4.3.3 Código

En este apartado hablaremos sobre la manera de actuar directamente sobre el código en ActionScript. Hay que tener en cuenta que no se va explicar de una manera muy profunda, si no que se hará un repaso de este explicando de una manera sencilla las posibilidades de configuración que nos ofrece. En este caso se ha trabajado sobre la versión Adobe Flash CS4.

4.3.3.1 Main

El código sobre el que trabajaremos principalmente será el de "Main.as" en ActionScript 3.0. Pero antes de empezar a trabajar con este, abriremos el archivo "EZFLAR.fla" con el programa Flash de Adobe. En este caso se ha utilizado la versión CS4 de Flash. Una vez abierto, buscaremos el botón de "Propiedades" que se encuentra en la parte superior derecha de la pantalla. Presionamos en él, y se abrirá el siguiente cuadro:

Una vez abierto el cuadro, en el campo "Clase" escribiremos "Main", de esta manera cada vez que ejecutemos "EZFLAR.swf" (el archivo ejecutable de Flash) buscará y cargará el archivo "Main.as".

Llegados a este punto, podemos empezar a tocar el código de "Main.as", representado en azul bajo estas líneas.

```
package {
	import com.tchatcho.EZflar;//tcha-tcho.com
	import flash.display.MovieClip;
	import com.transmote.flar.FLARmarkerEvent;

	public class Main extends MovieClip {
		private var _ezflar:EZflar;
		private var _symbols:Array = new Array();

		public function Main() {
			_symbols.push([["EZFLAR0.pat",                                    "text",
"12345678901234567890123456789012345678901234567890"],["mytext"]]);// 0
			_symbols.push([["EZFLAR1.pat", "Example_PNG.png"],["mypng"]]);// 0
			_symbols.push([["EZFLAR2.pat", "Example_JPG.jpg"],["myjpg"]]);// 0
			_symbols.push([["EZFLAR3.pat", "Example_GIF.gif"],["mygif"]]);// 0
			_symbols.push([["EZFLAR4.pat",                                    "url",
"http://www.google.com.br"],["mygoogle"]]);// 0
			_ezflar = new EZflar(_symbols);
			_ezflar.mirror();
			_ezflar.initializer(stage);
			_ezflar.onStarted(function():void {
				_ezflar.addModelTo([0,"Example_FLV.flv"], ["myflv"]);
				_ezflar.addModelTo([0,"twitter", "ezflar"], ["mytwitter"]);
				});
			_ezflar.onAdded(function(marker:FLARmarkerEvent):void {
				_ezflar.getObject(0,"mygif").rotationX = 90;
				trace(">>>>>>>>>>>>>> added: " + marker.marker.patternId);
```

```
                });
                _ezflar.onUpdated(function(marker:FLARmarkerEvent):void {
                        trace("["+ marker.marker.patternId+"]>>" +
                                "X:" + marker.x() + " || " +
                                "Y:" + marker.y() + " || " +
                                "Z:" + marker.z() + " || " +
                                "RX:" + marker.rotationX() + " || " +
                                "RY:" + marker.rotationY() + " || " +
                                "RZ:" + marker.rotationZ() + " || "
                        );
                });
                _ezflar.onRemoved(function(marker:FLARmarkerEvent):void {
                        trace(">>>>>>>>>>>>>> removed: " + marker.marker.patternId);
                });


                }
        }
}
```

En este apartado nos centraremos en la parte del código, que de una manera muy básica se puede manipular sin tener grandes conocimientos de programación.

De esta forma pasaremos directamente a la carga de markers y objetos sin pasar por la importación de librerías que se hace al inicio y por la creación de un array contenedor de markers y objetos. El código responsable de hacer esto el siguiente:

```
        public function Main() {
        _symbols.push([["EZFLAR0.pat","text","12345678901234567890123456789012345678901234567890"],["myt
ext"]]);// 0
        _symbols.push([["EZFLAR1.pat", "Example_PNG.png"],["mypng"]]);// 0
        _symbols.push([["EZFLAR2.pat", "Example_JPG.jpg"],["myjpg"]]);// 0
        _symbols.push([["EZFLAR3.pat", "Example_GIF.gif"],["mygif"]]);// 0
        symbols.push([["EZFLAR4.pat", "url", "http://www.google.com.br"],["mygoogle"]]);// 0
```

Más concretamente la función _symbols.push. Si observamos su estructura vemos que está compuesta por tres partes, la especificación del marker, en este caso EZFLAR0.pat, el objeto que queremos que aparezca cuando la cámara detecte nuestro marker, y por último el nombre con que identificar este objeto. Si observamos la sintaxis, tanto el objeto como el patrón están incluidos en unos mismos corchetes, entrecomillados cada uno de ellos y separados por una coma, para que nuevamente separada por una coma, entrecomillada y entre corchetes, aparezca el alias.

Como podemos observar, todos los archivos .pat, comienzan por el mismo nombre "EZFLAR", esto se debe a que para acceder a ellos se hace de una manera referencial por el último dígito de este, asignándoles en orden y comenzando en 0, números consecutivos. De esta manera y puesto que en el array que se crea al inicio de las líneas, conforme se va utilizando la función "push" y van incluyéndose todos los objetos de una manera ordenada, se asignan posiciones de igual manera. Por lo que, para nuestro proyecto cuando hagamos nuestro markers, les asignaremos el nombre de "EXFLAR" y les añadiremos la posición en la que se encuentran. A continuación, pondremos el nombre del objeto (incluida la extensión) que queramos que aparezca, y por último el alias que le queremos asignar. Así podemos incluir tantos objetos y marker como nuestro sistema soporte.

La segunda parte del código que tendremos en cuenta es la siguiente:

```
        _ezflar.onStarted(function():void {
```

```
_ezflar.addModelTo([0,"Example_FLV.flv"], ["myflv"]);
_ezflar.addModelTo([0,"twitter", "ezflar"], ["mytwitter"]);
});
```

En estas líneas, nos da la opción de asignar a un mismo marker a varios objetos, de esta forma podríamos hacer que al aparecer una imagen 3D, simultáneamente sonase una canción en mp3, todo ello con la función "addModelTo".

Para ello, podemos observar que la sintaxis de estas líneas es similar a la que teníamos al inicio, con la variante, que en lugar de aparecer el marcador, nos aparece un número. Esto es porque en lugar de especificarle un marker, le asignamos la posición en la que este se encuentra. De manera que si en la primera posición habíamos puesto una foto, en este apartado le asignaríamos un 0.

Por último, existen varias formas de hacer lo que se va a explicar a continuación, cambiar el posicionamiento y orientación de los objetos que queremos se muestren. Y esto se hace con el siguiente código.

```
_ezflar.onAdded(function(marker:FLARmarkerEvent):void {
_ezflar.getObject(0,"mygif").rotationX = 90;
trace(">>>>>>>>>>>>>> added: " + marker.marker.patternId);
});
```

El encargado de esto es la función "getObject", que nos da la opción de recuperar un objeto del array de objetos y utilizar diferentes cualidades de este para variar su posicionamiento. Como podemos observar en la sintaxis, pondríamos entre paréntesis el índice del marker al que hacemos referencia seguido de su alias, un punto y la característica que queremos variar. Como vemos en este código podemos variar la rotación con la sintaxis ".rotationX", que de igual manera nos permitiría variar la rotación en otros ejes sustituyendo la X por otra coordenada (Y,Z), siendo la rotación de grados. De igual manera podemos actuar sobre la escala de los objetos mediante la función "scale" siendo el resultado de la escala, el producto entre el número que pongamos y el tamaño real del objeto.

En este punto, conocemos los parámetros básicos de modificación del archivo "Main.as".

4.3.3.2 Control de audio

Como hemos comentado antes, existen otras opciones de modificación de los objetos en lugar de hacerlo en "Main.as". Una de las opciones es actuar directamente sobre el controlador del formato del objeto que queremos introducir, es decir, si vamos a introducir un archivo de audio mp3, actuaremos sobre el archivo de ActionScript que controla este formato directamente.

Para ello acudiremos a la carpeta donde se encuentran todos los formatos soportados: "src/com/tchatcho/constructors".

Comenzaremos con el archivo que controla el sonido mp3 "MP3constructor.as". El problema de este constructor es, que si queremos que a la vez que aparece un objeto en la pantalla suene un archivo en mp3, este constructor añade una imagen de un altavoz sonando. Para que esto no pase, tendremos que eliminar o comentar varias líneas de este constructor, concretamente, en este caso, quitaremos las siguientes líneas.

```
var request2:URLRequest=new URLRequest("com/tchatcho/constructors/soundicon.swf");
_loader.load(request2);
_loader.scaleY = 0.7;
_loader.scaleX = 0.7;
addChild(_loader);
var skinMaterial:MovieClip = this;
var front_material:MovieMaterial = new MovieMaterial(skinMaterial, true);
front_material.interactive = true;
front_material.animated = true;
front_material.doubleSided = true;
_front_plane = new Plane(front_material, 640, 640, 4, 4);
_front_plane.scale = 0.3;
this._universe.z = 3;
this._universe.x = -55;
this._universe.y = -55;
```

Una vez eliminadas estas líneas, cuando llamemos a un archivo de audio, se oirá el sonido pero no aparecerá nada más en la pantalla. Por otra parte en el archivo "oadingEZFLAR.as" también tendremos que borrar unas líneas para que no nos aparezca una animación de "loading..", mientras se reproduce la canción o fragmento de audio. Concretamente las líneas que tendremos que eliminar son las siguientes:

```
var noCamMsg:TextField              = new TextField();
noCamMsg.text                    = "LOADING\n...";
var format:TextFormat               = new TextFormat();
format.size                      = 19;
format.align                     = "center";
noCamMsg.setTextFormat(format);
noCamMsg.x = 1;
noCamMsg.y = 7;
this.addChild(noCamMsg);
*
var front_material:MovieMaterial    = new MovieMaterial(this, true);
front_material.doubleSided          = true;
var front_plane:Plane = new Plane(front_material, 400, 400, 2, 2);
front_plane.scale                   = 0.2;
front_plane.x                    = 1;
this._universe.addChild(front_plane);
```

4.3.3.2 Control de 3D

En este caso, solo existe un formato compatible con nuestro programa de realidad aumentada, los archivos de collada con extensión .dae (digital asset exchange). Este es un formato que define un esquema XML de estándar abierto para el intercambio de activos digitales entre diferentes aplicaciones interactivas en 3D. Así, si queremos que nuestra aplicación maneje alguna animación u objeto en 3D, tendremos que tener en cuenta que lo tendremos que exportar al formato .dae. De momento, existen varios programas de modelado 3D que son capaces de soportar este formato, como por ejemplo: 3Dmax (ColladaMax), Blender,... entre otros.

A lo que el código de control se refiere, lo podemos encontrar en la carpeta donde se encuentran todos los constructores "src/com/tchatcho/constructors", en el archivo DAEconstructor.as.

Básicamente las dos líneas que más nos interesan de este código serán:

```
this._mCollada.rotationZ = 270;
this._mCollada.scale = 0.5;
```

Como se ha comentado antes, existen varias formas de actuar sobre el control de los diferentes formatos que vamos a utilizar, en este caso, nos da la opción de actuar sobre la rotación y el tamaño directos de nuestros objetos 3D, nuevamente con las mismas funciones "rotationZ" y "scale".

Utilizaremos los cambios en este archivo cuando queramos que los cambios se efectúen sobre todos los archivos 3D que se vayan a cargar. De esta manera, si los cambios los hacemos en el "Main.as" los cambios podemos efectuarlos de forma localizada e individual para cada objeto.

4.3.3.3 Imagen y vídeo

Los archivos encargados de controlar estos formatos son PICTUREconstructor.as para imágenes y FLVconstructor.as para videos. En ambos casos los parámetros y funciones de modificación son similares.

El código de control es:

```
_front_plane = new Plane(pictureMaterial, 500, 500, 4, 4);
```

Esta línea se encarga del control del tamaño en píxeles tanto de las imágenes como de los videos. Tenemos que tener en cuenta que por defecto, cada uno de ellos viene con un tamaño diferente, que tendremos que adecuar según nuestras necesidades.

Por otro lado tenemos la orientación y el tamaño a escala de los dos objetos:

```
_front_plane.scale = 0.3;
```

Para escalar los objetos como si fuese un factor de multiplicación. Y

```
this._universe.rotationY = 0;
this._universe.rotationZ = -90;
```

para corregir la rotación de estos.

4.3.3.4 Control del programa

Comenzaremos hablando por "EZflar.as" el cual encontraremos en "src/com/tchatcho". En este archivo existen varios parámetros de configuración que podemos retocar a teniendo en cuenta nuestras necesidades.

Empezaremos por el control del tamaño de la ventana que se nos abrirá. Este código esta situado en la línea 40 de dicho archivo:

```
[SWF(width="640", height="480", frameRate="30", backgroundColor="#FFFFFF")]
```

Similar al control de video e imágenes, podemos configurar el tamaño de la ventana en píxeles (width=ancho, height=alto), que mostrará las imágenes capturadas por la webcam, así como número de imágenes por segundo que queremos que muestre. Tenemos que tener en cuenta que este no es el tamaño y ni las imágenes por segundo con las que

captura la webcam, sino la salida que le estamos dando a estas imágenes. Por defecto el frameRate (cantidad de imágenes por segundo) esta en 30, la cual es una buena cantidad y podemos dejar como está, por otro lado, el tamaño del visor de la webcam podemos cambiarlo dependiendo del tamaño del monitor y del navegador que vayamos a utilizar.

En caso de cambiar las dimensiones de la ventana, tenemos que tener en cuenta que en este mismo archivo no se ha utilizado una variable para asignar el alto y el ancho de la ventana, de manera que si lo hacemos en esta línea, para que todo siga funcionando correctamente, tendremos que cambiar los parámetros de las líneas sucesorias con el mismo valor con el que hemos colocado en esta.

Si seguimos adelante en el código, encontraremos las líneas que controlan en qué carpetas vas a estar situados tanto los objetos como los patrones:

```
private static const PATH_TO_MODELS:String = "models/";
```

Para indicar la carpeta donde se encontraran los objetos.

```
private static const PATTERN_PATH:String = "flar/patterns/";
```

Carpeta donde se encontrarán los marcadores .pat.

```
private static const PATTERN_RESOLUTION:uint = 16;
```

Número de bits con los que se codificará los patrones. Si recordamos, cuando creábamos un patrón, nos daba la opción de decirle el número de bits con que lo queremos codificar, de manera que este número tiene que ser igual o inferior a la cantidad con la que hemos codificado nuestros patrones, siendo la misma cantidad lo más aconsejable.

4.3.3.5 Control de la cámara

Concretamente el archivo que controla los parámetros de grabación de la webcam es “QRcodeReader.as” que se encuentra en la carpeta “src”. Concretamente en la línea 24 es donde podemos configurar la calidad con la que queremos que nuestra cámara graba:

```
camera.setMode(640, 480, 30);
```

Podemos intuir que el primer parámetro entre paréntesis será el ancho (o width que nos aparecía anteriormente) con un valor de 640 píxeles, separado por una coma encontramos el ancho (o heigth) y por último la cantidad de imágenes por segundo que queremos que grabe.

Nuevamente vemos comprometido el rendimiento de nuestra aplicación, ya que dependiendo de la potencia del equipo donde lo ejecutemos, tendremos más libertad para ampliar estos parámetros.

Esto se debe a que si no tuviésemos limitación alguna respecto al tamaño ni a la cantidad de imágenes por segundo que queremos que nuestra cámara grabe, a mayor tamaño de grabación, mayor detalle. De esta manera, le sería más fácil a nuestra aplicación encontrar los markers y a mayor número de imágenes por segundo, mayor precisión tendríamos. Como no es el caso, si aumentamos el tamaño en exceso de la grabación, el tamaño de las imágenes que tiene que procesar nuestra aplicación también aumenta, y a su vez el tiempo que requiere este en buscar los marcadores en esta imagen, si además le

añadimos un aumento de imágenes por segundo que procesar, nuestra aplicación dejaría de ser eficiente y sufriría un retardo a la hora de mostrar los objetos por pantalla.

4.3.4 Iluminación

La iluminación es un factor muy importante en este proyecto, y esto se debe a que dependemos de nuestra webcam y de la claridad con que esta perciba los marcadores.

Dicho esto, para la aplicación que vayamos a desarrollar funcione correctamente, deberemos preocuparnos de que los marcadores estén muy bien iluminados y de que los colores de estos sean, en lo posible, lo más blanco lo que tenga que ser blanco, y lo más negro lo que tenga que ser negro.

De manera que si queremos desarrollar una aplicación que vaya a estar expuesta al público, es posible que tengamos que contar con algún tipo de iluminación adicional, intentando a su vez que el material donde se encuentran los dibujos no sea reflectante, de lo contrario, los reflejos producidos por la iluminación podrían dificultar el reconocimiento de las imágenes.

5.- CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Realidad aumentada

Las posibles aplicaciones didácticas de la realidad aumentada pueden ser varias, puesto que esta consiste en añadir información virtual en tiempo real, en todos aquellos ámbitos en los que se necesite añadir información o dar soporte, podríamos considerar que la realidad aumentada podría servirnos de ayuda.

Los medios audiovisuales se han convertido hoy en día en elementos casi imprescindibles en la enseñanza, es por esto que, pizarras digitales y proyectores ya no son extraños de ver en las aulas de cualquier tipo de centro.

De esta manera la realidad aumentada da un paso más, y nos brinda la posibilidad de acercar la interactividad hasta el propio usuario. Hasta ahora, si un profesor quería mostrar el funcionamiento del corazón en movimiento en una clase de biología, mostraba un video de este. Con la realidad aumentada nos da la opción de que sea el alumno el que interactué con el sistema y sea él mismo capaz de manipular la animación, rotándola y acercándola como él quiera.

Según un estudio realizado a estudiantes del porcentaje aproximativo de los datos retenidos por estos, dependiendo de la actividad realizada (Sáenz y Más, 1979), estos fueron los resultados obtenidos.

10%	De lo que se lee
20%	De lo que se escucha
30%	De lo que se ve
50%	De lo que se ve y se escucha
70%	De lo que se dice y se discute
90%	De lo que se dice y luego se realiza

Como podemos observar en los porcentajes si somos capaces de que el alumno hable sobre el tema además de que luego "juegue" con algún tipo de aplicación relacionada con los conceptos que el profesor intenta transmitir, seremos capaces que incrementar al máximo este porcentaje de datos retenidos. Además de ser capaces de hacer que el aprendizaje en los colegios fuese más motivante para los alumnos, dando la posibilidad de jugar con el material haciendo que la materia impartida sea más fácil de asimilar.

Otra posible aplicación, con la inclusión de flash en la web y utilizando los nuevos dispositivos móviles que cuentan con conexiones a intenet de manera continua y con procesadores gráficos cada vez más potentes, podríamos hacer con ellos dispositivos con realidad aumentada dando la posibilidad de aportar información adicional de forma inmediata en restaurantes, universidades, museos, etc, sobre las novedades del día, o simplemente para dar una información de una forma diferente que nos aporte una cierta interacción con el usuario y aportándonos un toque de distinción frente a otros.

5.1.1 Símbolos

Por otra parte no debemos de olvidar los símbolos, parte fundamental en el aprendizaje. Puesto que nuestra aplicación de realidad aumentada consta de símbolos, se le debe dedicar cierta importancia a los dibujos que se introducen en los markers, ya que los símbolos son capaces de transmitir cierta información y ayudar así a la asimilación y mejor retención de conceptos. Esto se debe a que hay informaciones que se comprenden mejor mediante imágenes, y es que existen estudiantes que captan mejor las informaciones icónicas concretas que las verbales abstractas.

Es por ello que dependiendo del ámbito que le vayamos a ofrecer nuestra aplicación, deberemos de escoger unos símbolos u otros. De esta forma, las imágenes que utilicemos deberán ser claras, dando una idea clara y concisa de qué es lo que se va a representar con la misma, y todo ello para intentar reforzar los conceptos que se intentan transmitir.

5.2 Conclusiones generales

A lo que interacción se refiere, podemos concluir que los objetos y programas aquí utilizados, sí que pueden ser una buena herramienta de interacción para el entretenimiento. Concretamente en el caso de la utilización de Blender con la wii, puede ser una buena experiencia el configurar y desarrollar una pequeña aplicación para comprender mejor el funcionamiento su Game engine. Por otra parte, la posibilidad de desarrollar una aplicación más seria intentando darle una salida comercial, se ve limitada a la necesidad de poseer el wiimote. Es por esto que veríamos reducido su ámbito de utilización, a situaciones muy concretas.

Por otro lado, la tecnología de Wiiflash, aunque a primera vista puede dar a entender un sencillo y eficiente modo de utilización , cuando empezamos a utilizarlo, nos damos cuenta de su poca eficiencia, puesto que al poco rato de ejecutar nuestra aplicación, el ordenador queda bloqueado por una mala gestión de la memoria Ram. Por otra parte, si consultamos la página web donde dan soporte a esta aplicación, vemos que está sin actualizar desde 2008, siendo un breve post acerca de la compatibilidad de WiiFlash con Mac OS Leopard Snow, la única novedad de esta. Es por esto que directamente descartaría la posibilidad de desarrollar una aplicación con este programa. Al menos de una manera sencilla, ya que de tener los conocimientos necesarios podríamos intentar corregir el cógido causante del problema.

Finalmente con la realidad aumentada se puede concluir que de una manera sencilla, sin grandes conocimientos en programación, se puede desarrollar una aplicación con el único requerimiento de un ordenador y ciertos programas libres. En este caso, su posible adaptación a múltiples ámbitos es muy sencilla y únicamente requiere de una gran imaginación para poder hacer una experiencia más interactiva.

La ventaja de esta tecnología es que la mayoría del código utilizado, es código abierto, siendo GNU GPL la licencia mas habitual. De esta manera, podremos encontrar una gran variedad de códigos, que habitualmente estarán acompañados de múltiples explicaciones y comentarios, y todo esto respaldado por una gran comunidad como es la del software libre, pudiendo apoyarse en ella siempre que sea necesario.

6.- BIBLIOGRAFÍA

- Especificaciones técnicas y funcionamiento de Kinect:
http://www.dimensionxbox.com/noticias/kinect/conoce-con-especificaciones-tecnicas-y-a-fondo-a-kinect-469

- Especificaciones técnicas y funcionamiento de PlayStation Move:
http://www.hardgame2.com/ps3/articulo-4952especial-playstation-move-review-hardware.html&page=2

- Proyecto 6th Sense:
http://www.pranavmistry.com/projects/sixthsense/

- Microsoft Surface:
http://www.microsoft.com/surface/en/us/default.aspx
http://www.genbeta.com/windows/microsoft-surface-en-espana-lo-probamos

- Wiimote:
http://www.taringa.net/posts/info/1843549/Como-funciona-el-Wiimote.html
http://www.broadcom.com/press/release.php?id=854744

- Protocolo OSC:
http://www.taringa.net/comunidades/cienciainformatica/633465/Introducción-al-OSC-(Open-Sound-Control)---Primer-parte.html

- Blender:
http://www.blender.org/

-Artoolkit:
http://www.hitl.washington.edu/artoolkit/

- EZFlar:
http://www.ezflar.com/home/show_home